Bachelor of Science Honors Thesis

Fast Back-projection Algorithm for Synthetic Aperture Radar Imaging System with a Lightweight Unmanned Aircraft System

By Jingong Huang

July 12, 2018

Dept. of Electrical and Computer Engineering

The Ohio State University

Advisor Lee Potter

Abstract

The back-projection algorithm coherently combines backscatter or tomography data to produce images. It has been widely used in the medical imaging field and for synthetic aperture radar imaging. With back-projection, the imaging process can be very time-consuming; therefore, in this effort we have presented a fast back-projection algorithm that provides significant computational savings with only modest loss in image quality. This paper introduces the back-projection algorithm and two fast back-projection algorithms. In addition, this paper discusses an IQ balance method necessary in practice for calibration of hardware imperfections in a quadrature receiver.

Table of Contents

List of Figures:

List of Tables

Chapter 1: Problem Description

In most lightweight unmanned aircraft systems, optical cameras are used to take images. However, optical imaging techniques have operation condition limitations. The 2018 Synthetic Aperture Radar (SAR) research team built an imaging radar that can operate on a lightweight unmanned air system (UAS). The imaging capability can provide vision when optical systems may fail, either at night, through smoke, or through the fog. The new technology could be useful for emergency response or security applications. An example ideal image of three reflectors is shown in Figure 1. The image resolution is limited by the bandwidth of the radar transmission and the sector of angles subtended by the synthetized aperture. Interest in the potential technology is evidenced by the equipment funds provided by local industry.

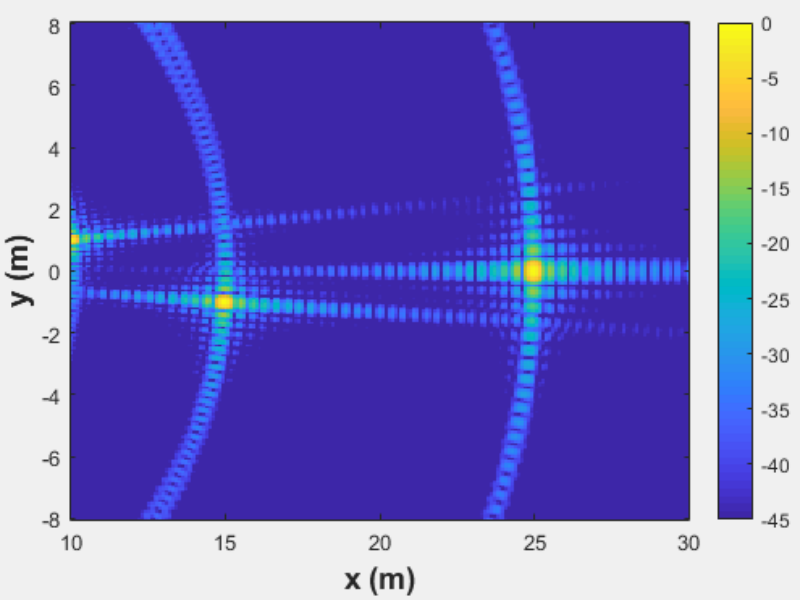


Figure 1: The ideal simulated image

The back-projection algorithm has been widely used in Synthetic Aperture Radar (SAR) imaging systems [2]. The traditional Back-projection algorithm has a large time cost when the image has high resolution, or when the image is constructed using large data sets. The color of each pixel in the image represents the signal intensity. The signal intensity is the summation of all the signal intensity from every SAR system

location along the flight path [5]. Therefore, to speed up the image forming process, the team used a fast Back-projection algorithm on the SAR imaging system. This technique may be used in real-time SAR systems in the future.

Chapter 2: Background

Section: 2.1: Overall introduction of the SAR project

Synthetic Aperture Radar (SAR) was implemented on a lightweight drone to provide large area vision with high resolution without the optical limitation. By forming a large synthetic array, radar pulses are transmitted along the platform's flight path to construct SAR image [2]. An image can be formed from the received echoes by solving a large set of linear equations relating echo times to distances. Back-projection provides an approximate least-squares solution to this very, very large system of equations.

The undergraduate research project reported in this thesis is a portion of a larger effort to build and demonstrate a lightweight airborne imaging system. One critical system requirement is the ability to form a high-resolution image when optical imaging fails. The SAR system block diagram is shown in Figure 3. The SAR system mainly contains four parts: a radar transceiver, Raspberry Pi control unit, GPS modules, and a drone (refer to Figure 2 and 3).

Figure 2: SAR system

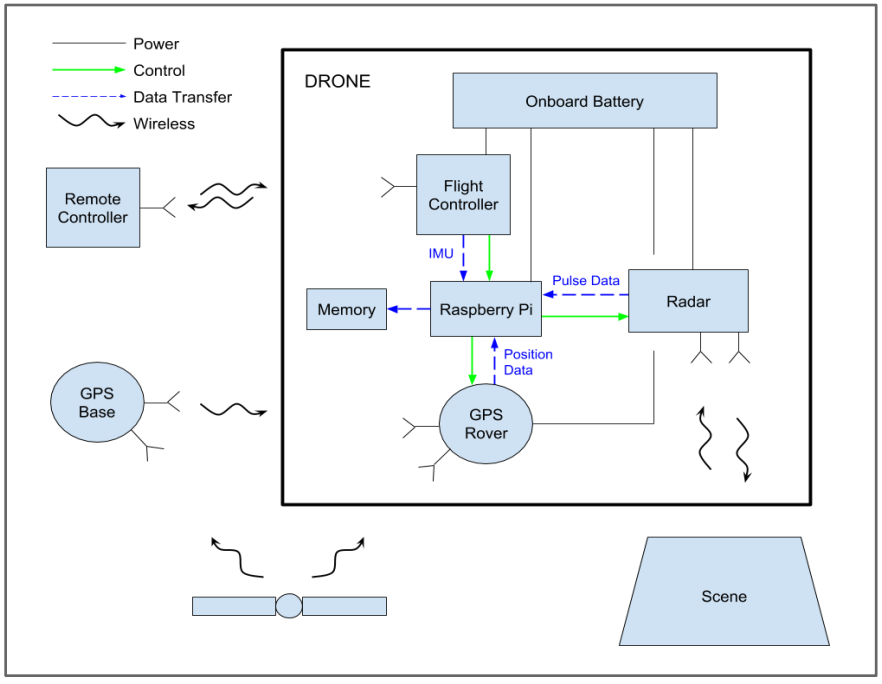


Figure 3: System block ***diagram*** *[8]*

Section: 2.2: Teamwork

The project team is divided into four groups as shown in Table 1. In this project, I provided four contributions. First, I adapted existing back-projection algorithms to create SAR images using the radar echoes and GPS data recorded on the UAS,. Second, I developed and implemented a simulator for software testing of SAR imaging scenarios. The first and second contributions were developed jointly with my team partner, Shiqi Yang. Third, I implemented an IQ balance program to correct the DC and phase offsets of the IQ data; these errors arise due to hardware imperfections in the quadrature demodulator. Fourth, I devised and implement a fast-approximate algorithm for near real-time imaging.

Group and group members	Responsibility
Platform: Joy Smith, Sarah Greenbaum	Airframe, flight control, wireless control, experiment setup
GPS: Brandi Downs	GPS postponing, Drone flight
Radar: Daniel Wharton	Kalman Filter
Imaging software: Shiqi Yang and Jingong Huang	Data collection and image processing including Back-projection, Auto-focus, and fast calculation algorithm

Table 1: Subgroups and Tasks

There are two ways to achieve fast back-projection [1]. The more common way is that the final image is divided into several parts to form sub-images. Each sub-image is constructed individually. Finally, all the sub images are added together to make a new image. The second way is to create several low-resolution sub-images. The low-resolution images can then be added together to create a final image. The second method was applied to this project. I also developed a new method to speed up the image construction process.

Section: 2.3: FAA, FCC, and OSU Regulation

The whole project satisfied the requirements of the FAA and FCC regulations: 14 CFR Part 107, 47 CFR Sections 15.205 and 15.521 (refer to Table 2). The drone was registered with the FAA, and the pilot Brandi Downs obtained a Remote Pilot Airman Certificate for a small UAS. To satisfy the OSU regulation, the SAR team has submitted an UAS Request Form and an Export Review Request Form, and permission has been granted. The radar transceiver unit is operated on the band 9.6 to 10.0 GHz, in compliance with relevant regulations; operating the transceiver at a bandwidth greater than 500 MHz would subject it to FCC regulations on UWB radiators, which include a prohibition against ultra wide-band radiators on any aircraft.

Component	Application	Regulation Limit	Plan for Compliance	Regulating Body
UAS	Flight Altitude	< 400 ft	Fly under 400 ft	FAA
UAS	Total Weight	< 55 lbs	< 10 lbs	FAA
Radar	Bandwidth	< 500 MHz	< 500 MHz	FCC
Radar	Restricted Frequency Bands	9.3 - 9.5 GHz 10.6 - 12.7 GHz	Operate within 9.55 - 10.05 GHz	FCC

Table 2:FAA and FCC regulation

Chapter 3: Algorithm Description

Section: 3.1: Back-projection

The back-projection algorithm uses data gathered from a radar system. The team chose to use the Ancortek Radar 980 AD module. The Ancortek center operating frequency was 9.8 GHz with a bandwidth of 400 MHz. The collected echoes were in the frequency domain in the form of IQ data. By plotting the magnitude of a single echo in one dimension, the graph showed peaks which represented signal strength of the detected reflector. A Fourie transform converted the sampled IQ data to reflection versus distance (refer to Figure 4); then, each echo was mapped to every pixel of the image by adding the complex-valued echo value at a given distance to all pixels in the image at that distance from the transceiver location. The back-projection algorithm repeats this process for the signals received at each location (refer to Figure 5). Finally, the algorithm sums the mapped values in each pixel, resulting in a two-dimensional image (refer to Figure 1) [2][5].

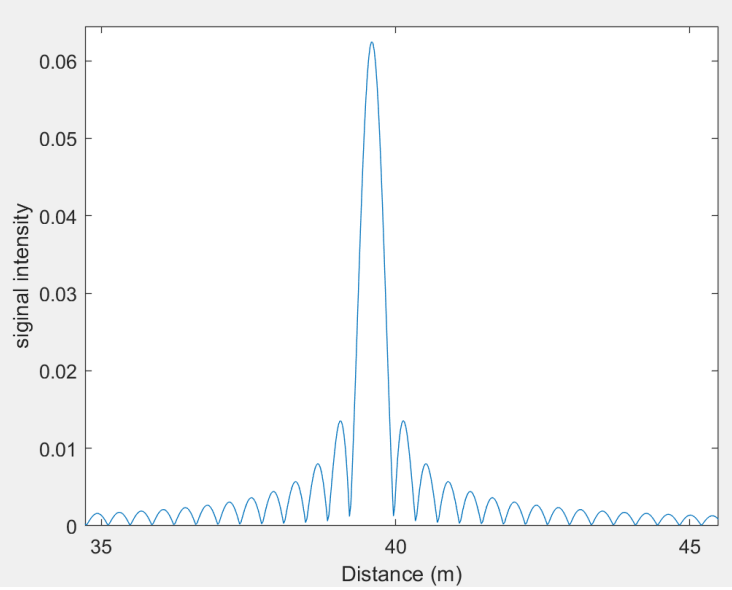


Figure 4: A single echo in one dimension is given by the Fourier transform of the received IQ data

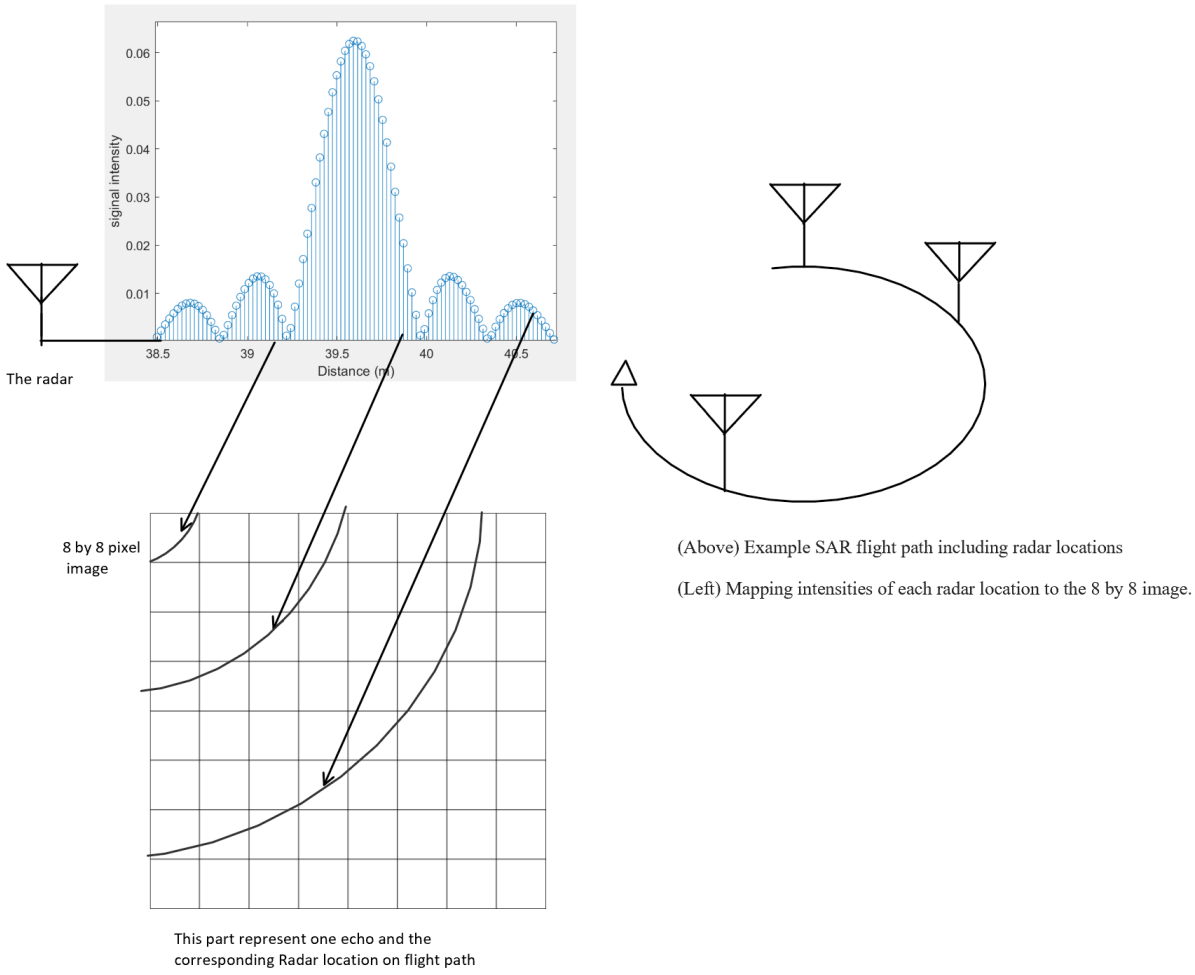


Figure 5: Mapping samples to image grid

Section: 3.2: Fast back-projection

For an $N \times N$ image, the fast back-projection reduces computation cost from

$Operation = O(N^3)$ to $Operation = O(N^2 log_2 N)$ through a divide and conquer approach [1], [4], [10]. Computation costs come from range calculation, interpolation, and summation operation times.

The fast back-projection contains three main steps [6]. First, the algorithm separates the phase history into M parts (refer to Figure 6). Then the subparts of the phase history are used to form sub-images through a normal back-projection algorithm. Finally, the sum of the M sub-images is used to make an image (refer to Figure 7). The way to construct the final image is shown in figure 7. The M value in Figure 6 and Figure 7 are equal to 2 and the operation cost is $\frac{N^3}{4}$ (refer to equation 1).

$$\text{Equation 1: Operation} = 2 \times \frac{N}{2} \times \frac{N^2}{4} = \frac{N^3}{4}$$

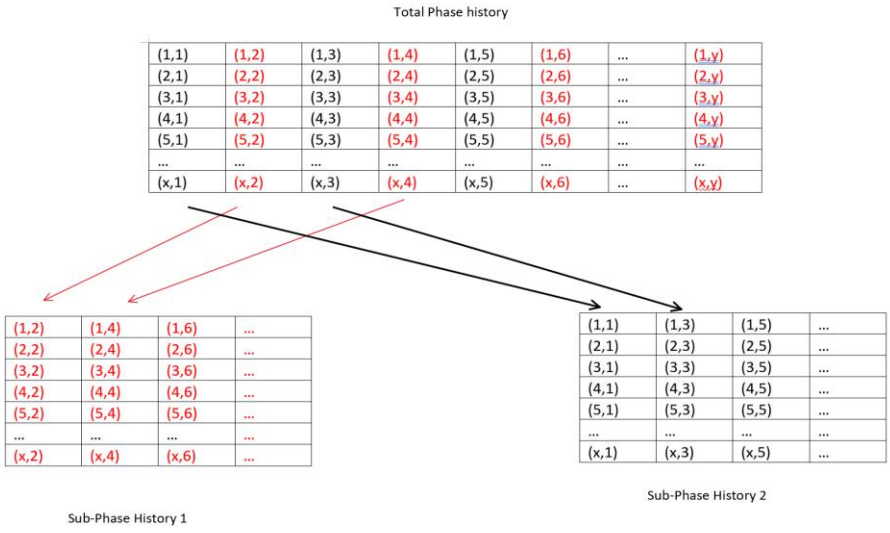


Figure 6: Divided Phase History [6]

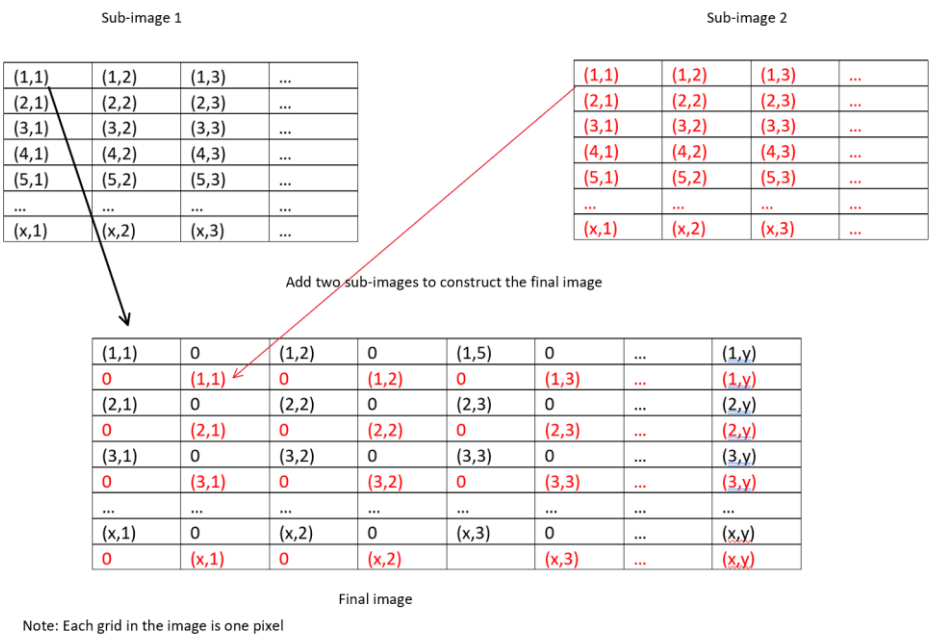


Figure 7: Construct final image [6]

Compared to the traditional back-projection algorithm, this method can construct an image more quickly, but compromises image quality. Therefore, a new method was developed to construct a good quality image with fast construction speed. It provided better image quality than fast back-projection and took less time to construct images than previous implementations of the back-projection algorithm (refer to Figure 8). The operation cost of this algorithm is $\frac{N^3}{2}$ (Refer to Equation 2).

$$Equation\ 2: Operation = 2 \times N \times \frac{N^2}{4} = \frac{N^3}{2}$$

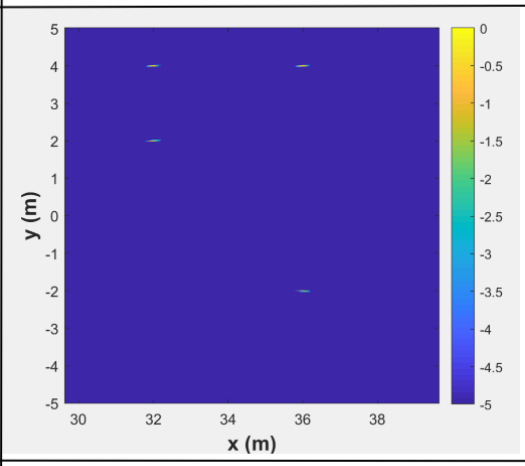
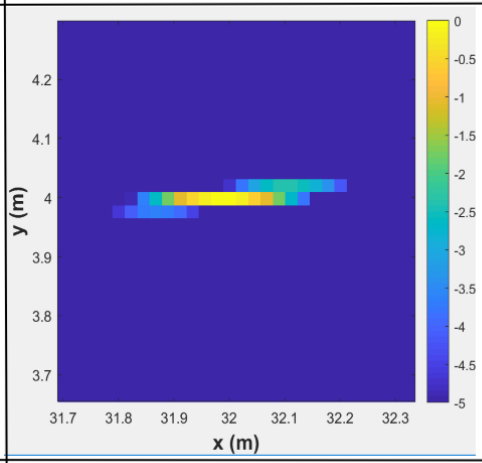
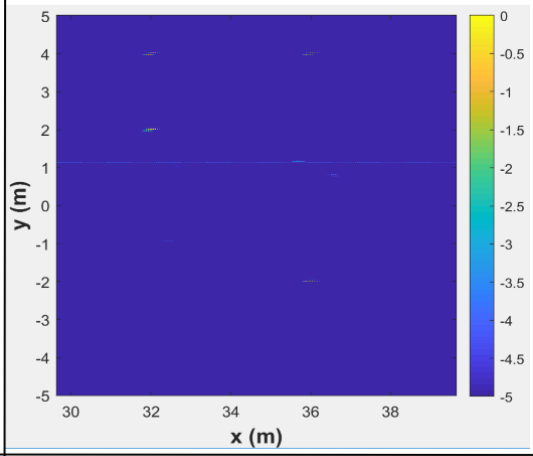
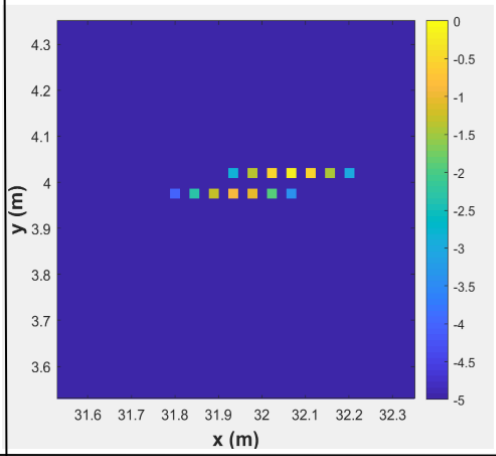
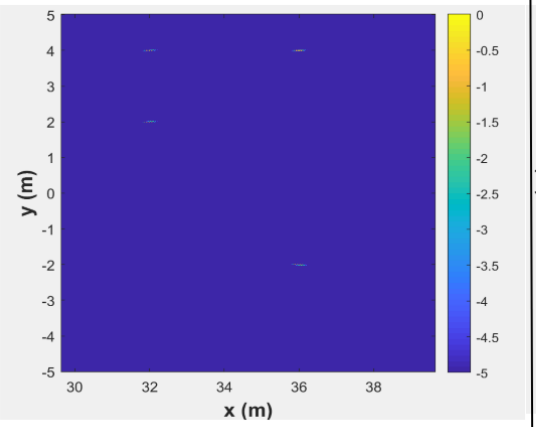
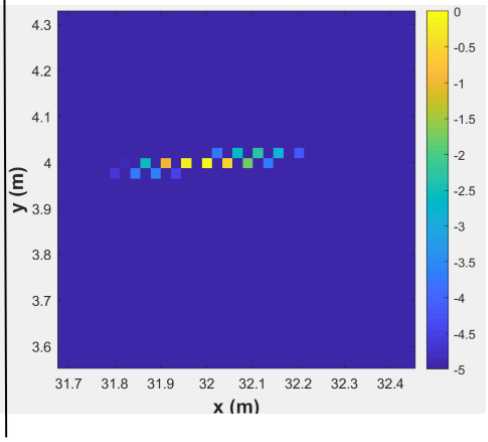
	Overall Image	Zoom in First object
Back-projection: Time cost is 5.997866 seconds.		
Fast Back-projection : Time cost is 0.918428 seconds.		
New Fast Back-projection : Time cost is 2.194623 seconds.		

Figure 8: Different image quality and time cost

Instead of dividing phase history into several parts, the new fast back-projection algorithm separated the image grid used in the back-projection algorithm. Then the main function called

the back-projection function M times to form M sub-images. Finally, these sub-images were added to construct a final image (refer to Figure 7).

Fast factorized back-projection algorithm is another image reconstruction method that has been widely applied to SAR imaging [10] [11]. Fast factorized back-projection (FFBP) contains two steps with a polar format for pixels. First, the algorithm divides the aperture and constructs sub-images. Each sub-aperture allows construction of a coarser image with one half as image pixels in the angle dimension (refer to Figure 9). Second sub-images are added with necessary phase factor and interpolation to make a finer resolution image at the next stage (refer to figure 10). The aperture partition can be repeated for up to $log_2 N$ stages. The operation cost of fast factorized back-projection algorithm is $O(N^2 log_2 N)$. The saving is negligible for small image but can be two orders of magnitude for large images, e.g. With 8000 or more pules.

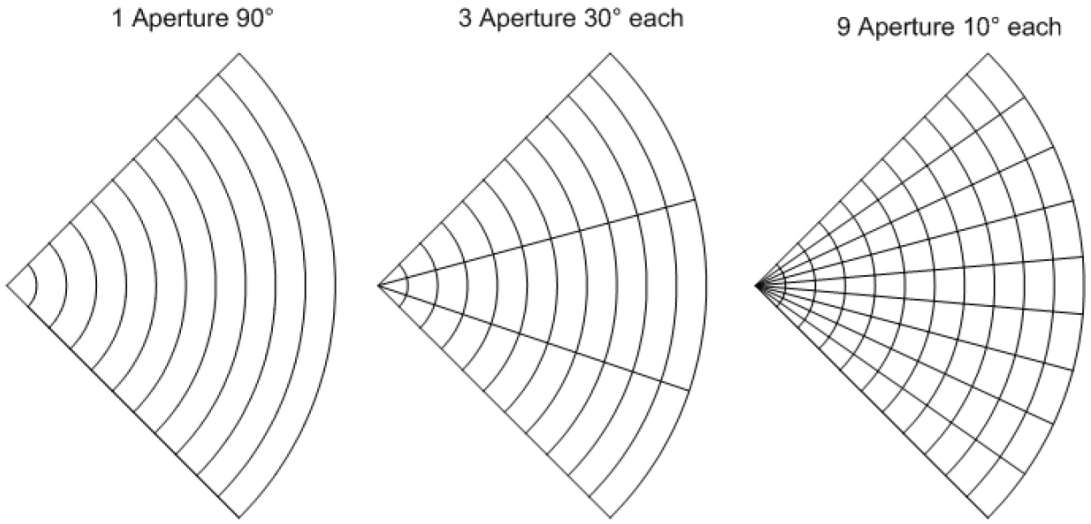


Figure 9: The relationship between aperture angles integration number [10]

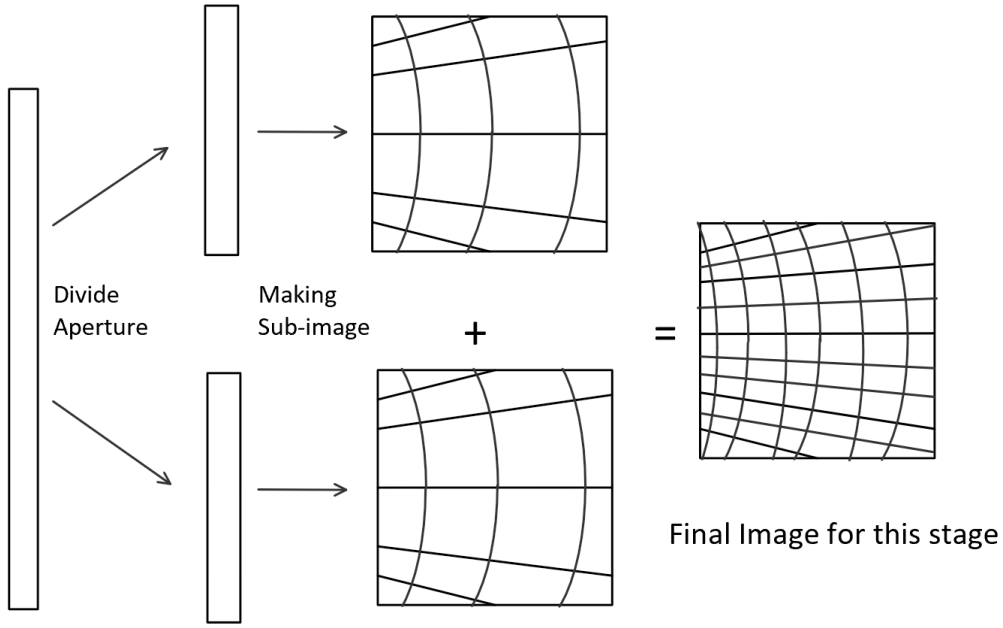


Figure 10:Process of fast factorized back-projection shown for one level of factorization

A third fast back-projection approach is similar to FFBP but decimates range data at each stage to also reduce the number of pixels in the range direction at each stage. This yields a quad-tree approach [12] [13].

Section 3.3: IQ balance

The IQ balance algorithm [9] was used as one of the functions in the back-projection code to provide better image quality. This IQ balance algorithm is used to correct the collected IQ data by SAR system. The IQ imbalance causes ghost peaks in conjugate symmetric location within the periodic 1D range line.

IQ imbalance in radar transceivers is due to hardware imperfections. The IQ data collection can be expressed by a block diagram as shown in Figure 11. Because of the phase error and DC offset, the waveform changed from $\mathrm{I(t)} = \mathrm{A} \times \cos(\omega \mathrm{t})$ and $\mathrm{Q(t)} = \mathrm{A} \times \sin(\omega \mathrm{t})$ to $y_I(t) = A \times \cos(\omega t) + B_I$ and $y_Q(t) = A \times \beta \times \sin(\omega t + \varphi) + B_Q$.

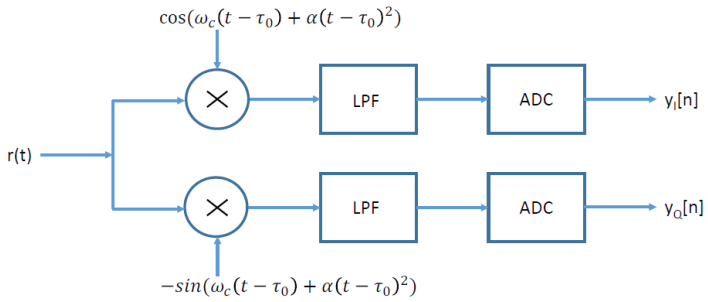


Figure 11: IQ data collection process [9]

Let y'_I be the average of $y_I(t)$ and y'_Q be the average of $y_Q(t)$. The trigonometric identity function $\sin(\mathrm{a} + \mathrm{b}) = \sin(\mathrm{a}) \times \cos(\mathrm{b}) + \cos(\mathrm{a}) \times \sin(\mathrm{b})$ can be used to create the IQ matrix as shown in Equation 8 below.

By substituting collected data into the function, the corrected data was produced (refer to Equations 3 to Equation 10) [9]. Figure 12 is a test of IQ balance function. The function could correct the raw data and made the corrected data match the ideal data. For radar sweep data, an alternative to Equations 3-9 for the IQ balance approach is given in [7]

Equation 3:
$$\mathrm{VarI} = \frac{1}{T}\int_0^T (y_I(t) - \bar{y_I})^2 dt = \tfrac{1}{2}A^2$$

Equation 4:
$$\mathrm{VarQ} = \frac{1}{T}\int_0^T (y_Q(t) - \bar{y_Q})^2 dt = \tfrac{1}{2}A^2\beta^2$$

Equation 5:
$$\begin{aligned}\mathrm{CovIQ} &= \frac{1}{T}\int_0^T (y_I(t) - \bar{y_I})(y_Q(t) - \bar{y_Q})dt \\ &= \frac{1}{T}\int_0^T A\cos(\omega t)\left\{A\beta\sin(\omega t)\cos(\phi) + A\beta\cos(\omega t)\sin(\phi)\right\}dt \\ &= \tfrac{1}{2}A^2\beta\sin\phi\end{aligned}$$

Equation 6:
$$\frac{\mathrm{VarQ}}{\mathrm{VarI}} = \beta^2$$

Equation 7:
$$\frac{\mathrm{CovIQ}}{\mathrm{VarI}} = \beta\sin\phi$$

Equation 8:
$$\sqrt{\frac{\mathrm{VarQ}}{\mathrm{VarI}} - \left(\frac{\mathrm{CovIQ}}{\mathrm{VarI}}\right)^2} = \beta\sqrt{1-\sin^2\phi} = \beta\cos\phi$$

Equation 9:
$$\frac{-\beta\sin\phi}{\beta\cos\phi} = -\tan\phi$$

Equation 10:
$$\begin{bmatrix} I(t) \\ Q(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\tan\phi & \frac{1}{\beta\cos\phi} \end{bmatrix} \begin{bmatrix} y_I(t) - \bar{y_I} \\ y_Q(t) - \bar{y_Q} \end{bmatrix}$$

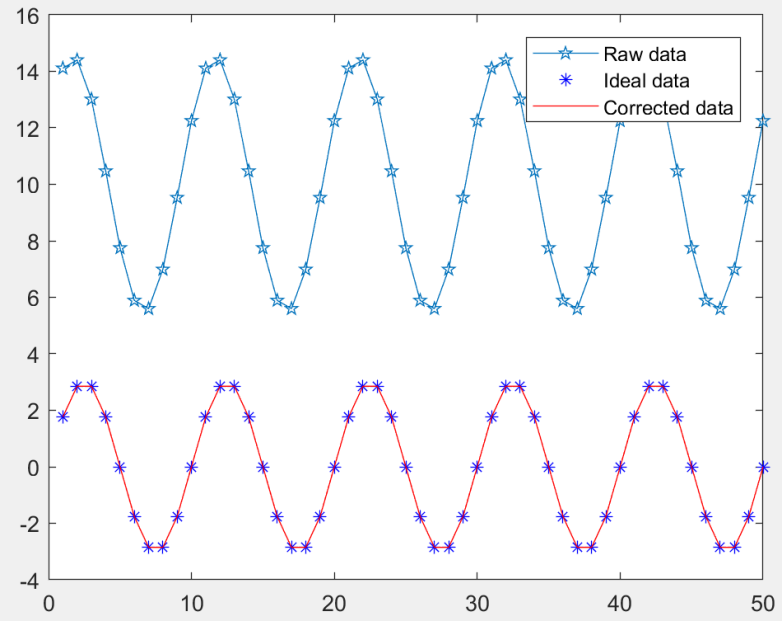


Figure 12: IQ balance test example

Chapter 4: Experimental Results and Analysis

Section 4.1: Experimental results

Testing was done using an Ancortek Radar system with three reflectors on a soccer field (refer to Figure 13). A rail was used to collect data at 2cm intervals at 41 locations across a total aperture length of 0.8m. Reflectors were placed with the center of the aperture at (0,0) (refer to Table 3 for locations). The bandwidth was $400 * 10^6\ Hz$ with the lowest frequency of each pulse at $9.6 * 10^9\ Hz$. The number of samples per sweep was 1024.

Figure 13:Radar system test with rail

Object	(X,Y,Z) m
1	(25, 0, 0.98)
2	(10, 1, 0.98)
3	(15, -1, 0.98)

Table 3: Location of three reflectors

An ideal simulated image was created with the back-projection algorithm on simulated data (refer to Figure 14). The reflector location in the image can be matched with known reflector locations.

The collected data were used to create 100 1-D range lines from 100 pulse to test the accuracy of the experiment (refer to Figure 15). The three peaks in the image corresponded to the three reflectors.

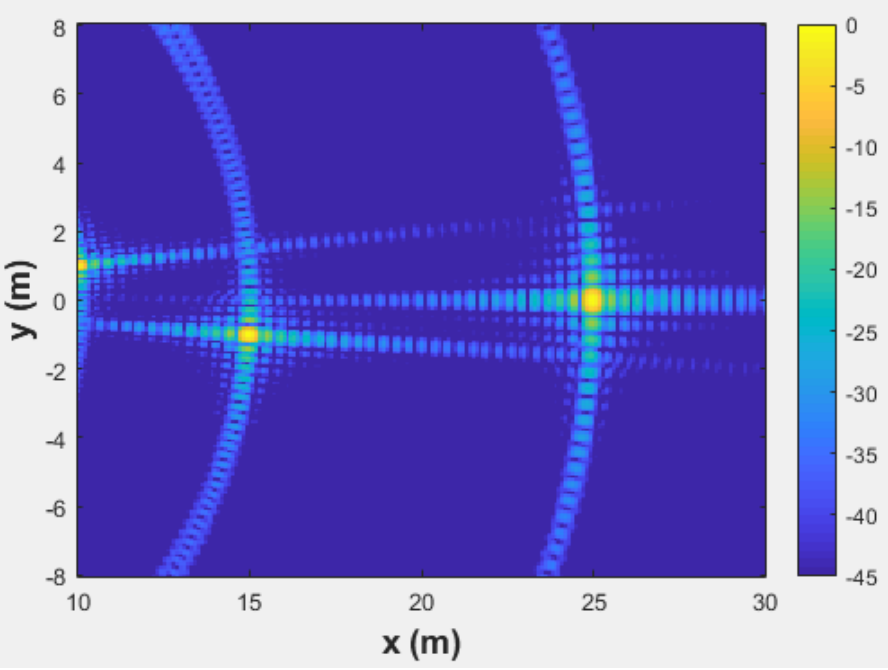


Figure 14: Simulated image of three reflectors

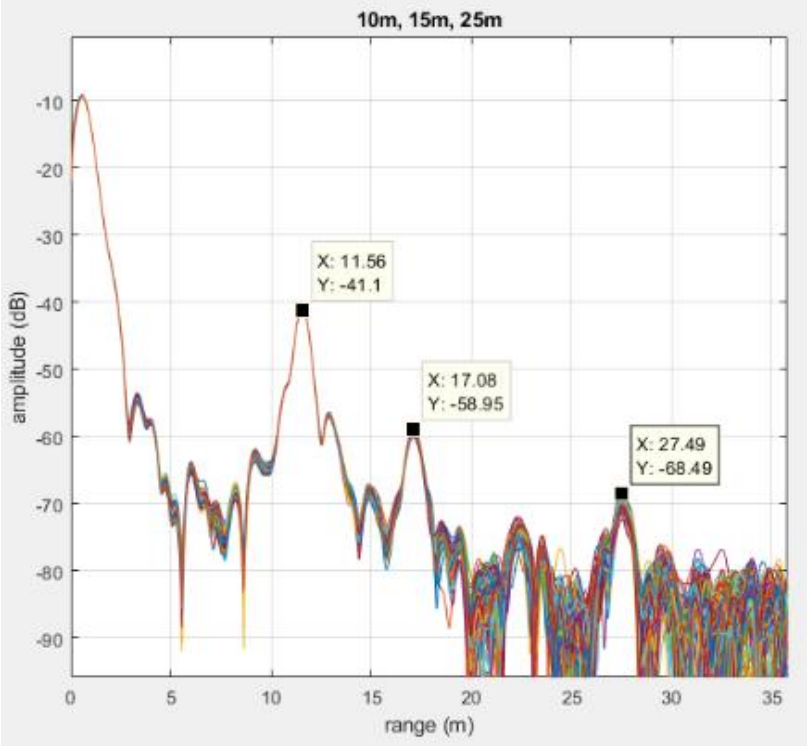


Figure 15: 1D image of the three reflectors

Figure 16 is the experimental 2D Back-projection image with three distinguishable reflectors. Constructing this 2D image with the back-projection algorithm took 22.809 seconds. Figure 17 is the 2D image of the three reflectors constructed by the new fast back-projection algorithm. This image formation took 17.223 seconds.

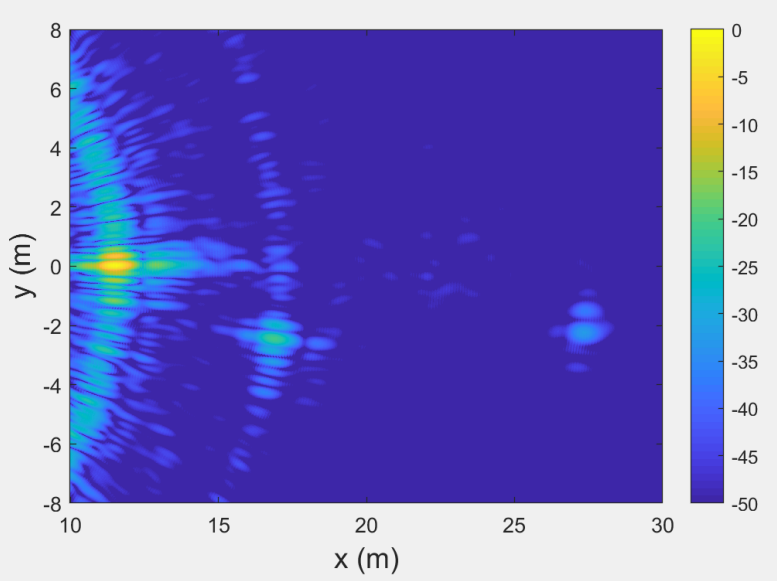


Figure 16: Three-reflector image constructed by back-projection algorithm

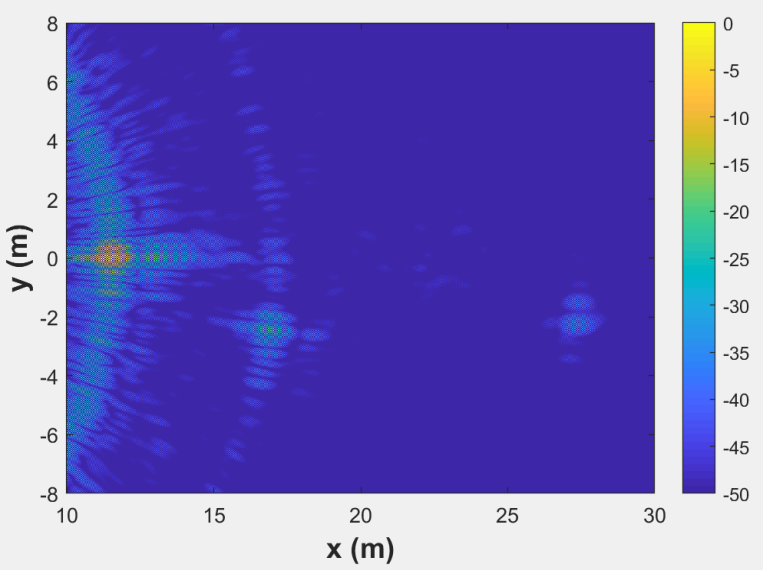


Figure 17: Three-reflector image constructed by new fast back-projection algorithm

Section 4.2: Experiment result analysis

	Delta X Expected	Delta X Image	Error	Delta Y Expected	Delta Y Image	Error
Reflector 1 to 2	5.00	5.50	0.50	-2.00	-2.56	0.56
Reflector 2 to 3	10.00	10.40	0.40	1.00	0.32	-0.68
Reflector 1 to 3	15.00	15.90	0.90	-1.00	-2.24	1.24

Table 4: Experiment result analysis

Experimental measurement of the reflector locations is given in Table 4. The average amount of error in X and Y distances relative to expected values is 0.71m. The error may have been caused by a hardware offset, R_0, or inaccuracies in the linear chirp causing errors in the delta f parameter which affected the output of the equation below. [2]

$$Equation\ 9: Range\ location = R_0 + n * \frac{c}{2} * \frac{1}{Delta\ f}$$

In the equation above, n is the FFT sample number, c is the speed of light, and R_0 and delta f are shown in Figure 18.

Errors in R_0 and $Delta\ f$ are related to the physical device. Figure 18 is the ideal situation. With precisely known $Delta\ f$, after taking samples from the sweep, value of the $Delta\ f$ contain error. The equation can be rewritten as the following:

$$Equation\ 10: Range\ location = R_0 + n * \frac{c}{2} * \frac{1}{Delta\ f} + n * \frac{c}{2} * error$$

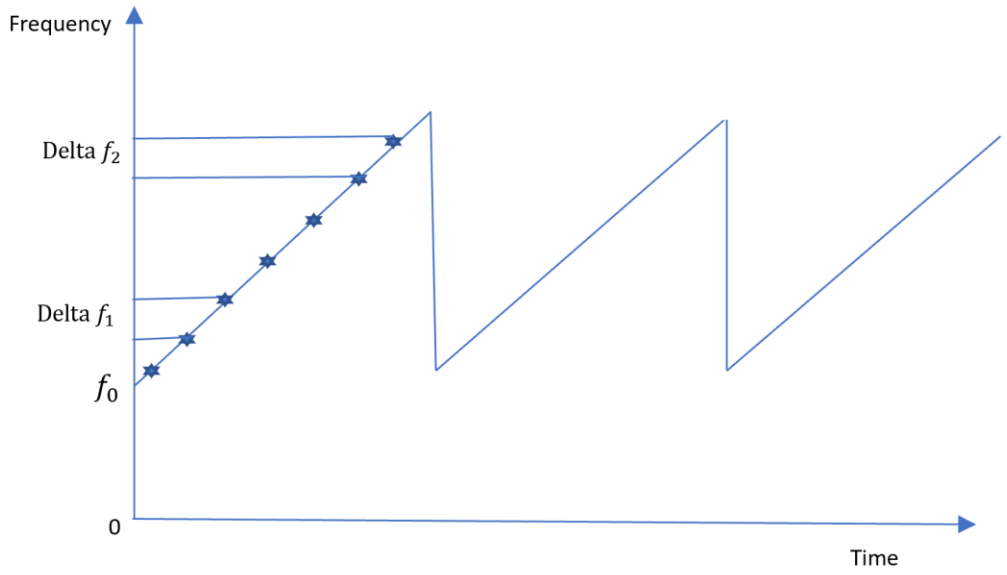


Figure 18: Ideal collected radar sweeps

Expected (m)	Range Line (m)	Difference from Expected (m)
7.00	8.17	1.17
8.00	9.19	1.19
9.14	10.81	1.67
10.05	11.60	1.55
10.05	11.33	1.28
10.20	11.61	1.42
15.03	17.19	2.16
27.43	30.02	2.59
36.58	39.89	3.31
54.86	59.03	4.17

Table 5: Various experimental results

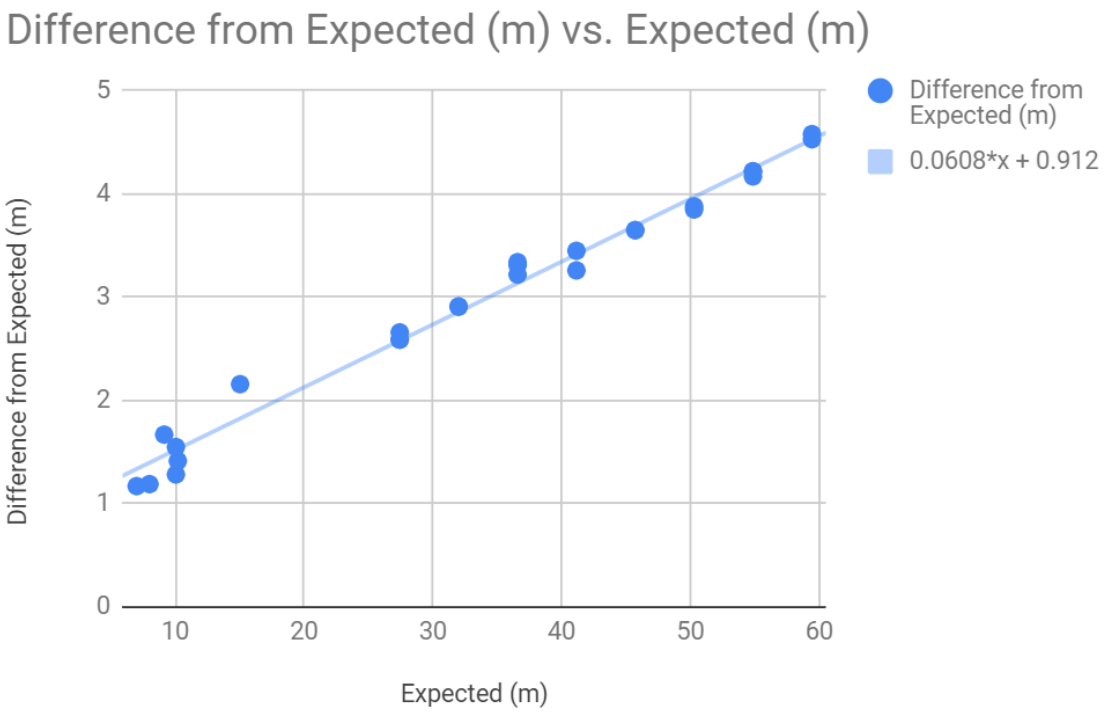


Figure 19: Error analysis graph

Based on different experiment data with different locations and reflectors, it was determined that the distance error increased linearly with reflector distances from the radar (refer to Figure 19 and Table 5). In summary, errors in R_0 and $Delta$ f are conjectured to arise from the physical transceiver device and are consistent with the linear fit experimentally observed in Figure 19. Corrections to these parameters are easily incorporated into the imaging algorithm.

Chapter 5: Conclusion

Three image reconstruction algorithms were used in the SAR project, all using Matlab. The SAR system successfully detected targets using each algorithm. A high-resolution 2D SAR image was constructed. The team produced images with three different levels of quality and image construction speeds to satisfy the requirement of the system user. Based on the simulation, experiment, and discussion of three SAR image construction algorithms, a tradeoff between image quality and image construction speed was found. High-quality SAR images cost more time to construct. When more sub-images were used in the fast back-projection, the time cost was less, and the image quality was worse. Image reconstruction included calibration of practical non-ideal characteristics of the transceiver: IQ imbalance, stretch processing delay, and linear FM chirp rate.

References

[1] Daniel E. Wahl, David A. Yocky, Charles V. Jakowatz, "An implementation of a fast backprojection image formation algorithm for spotlight-mode SAR," Proc. SPIE 6970, Algorithms for Synthetic Aperture Radar Imagery XV, 69700H (15 April 2008); doi: 10.1117/12.779401

[2] Charles V. Jakowatz, Daniel E. Wahl, David A. Yocky, "Beamforming as a foundation for spotlight-mode SAR image formation by backprojection," Proc. SPIE 6970, Algorithms for Synthetic Aperture Radar Imagery XV, 69700Q (15 April 2008); doi: 10.1117/12.779305

[3] L. Demanet, M. Ferrara, N. Maxwell, J. Poulson, and L. Ying, "A butterfly algorithm for synthetic aperture radar," Proc. SPIE 8051, Algorithms for Synthetic Aperture Radar Imagery XVIII, 805105 (4 May 2011); doi:10.1117/12.888948

[4] S. I. Kelly and M. E. Davies, "A fast decimation-in-image back-projection algorithm for SAR," *2014 IEEE Radar Conference*, Cincinnati, OH, 2014, pp. 1046-1051. doi: 10.1109/RADAR.2014.6875748

[5] L. A. Gorham and L. J. Moore, "SAR image formation toolbox for MATLAB," Proc. SPIE 7699, Algorithms for Synthetic Aperture Radar Imagery XVII, 769906 (18 April 2010); doi:10.1117/12.855375

[6] S. U. Danish, "DEVELOPMENT AND IMPLEMENTATION OF A FAST FACTORIZED BACKPROJECTION CODE TO SPEED UP IMAGE RECONSTRUCTION FOR SYNTHETIC APERTURE RADAR," BS Honors Thesis, College of Engineering of The Ohio State University, 2011.

[7] N. Sivannarayana and K. V. Rao, “I-Q imbalance correction in time and frequency domains with application to pulse doppler radar,” Sadhana, vol. 23, no. 1, pp. 93–102, 1998.

[8] Downs. Brandi, Pycraft. Aaron and Smith. Luke “Synthetic Aperture Radar on an Unmanned Aircraft System: Final Report,” ECE 4901 Report, Ohio State University, May 2018.

[9] "I/Q Correction [Analog Devices Wiki]", Wiki.analog.com, 2018. [Online]. Available: https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms1-ebz/iq_correction. [Accessed: 16- Jul- 2018].

[10] H. Andreas and J. Lars, “FAST FACTORIZED BACK-PROJECTION IN AN FPGA,” Master Thesis, School of Information Science, Computer and Electrical Engineering Halmstad University, January 2006.

[11] L. M. H. Ulander, H. Hellsten and G. Stenstrom, "Synthetic-aperture radar processing using fast factorized back-projection," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 3, pp. 760-776, July 2003. doi: 10.1109/TAES.2003.1238734

[12] McCorkle, John & Rofheart, Martin. (1996). Order N^2 Log(N) backprojector algorithm for focusing wide-angle wide-bandwidth arbitrary-motion synthetic aperture radar. doi: 10.1117/12.243085.

[13] LeRoy A. Gorham, Uttam K. Majumder, Peter Buxa, Mark J. Backues, Andrew C. Lindgren, "Implementation and analysis of a fast backprojection algorithm", Proc. SPIE 6237, Algorithms for Synthetic Aperture Radar Imagery XIII, 62370G (17 May 2006); doi: 10.1117/12.674024

Chapter 6: Appendices.

Section 6.1 Back-Projection Main function MATLAB code

```
close all, clear, clc, format compact
tic
c = 299792458; % Light speed
simdata.BW = 400 * 10^6; % BandWidth
simdata.minF = 9.6 * 10^9; % Lowest Frequency of each pulse
simdata.K = 256; % Number of samples per sweep
simdata.freq = linspace(simdata.minF, simdata.minF+simdata.BW, simdata.K); % Frequency of the chirp
simdata.deltaF = simdata.freq(2)-simdata.freq(1);
simdata.Nfft = 4096;
%image setup
simdata.CenterX = 34.64;
simdata.CenterY = 0;
simdata.Wx = 10;
simdata.Wy = 10;
simdata.Nx = 450;
simdata.Ny = 450;
dyn_range = 5;              % dB of dynamic range to display
%=============================================
%                   Object Location
Object(1) = sim_setObject(32, 4, 0);
 Object(2) = sim_setObject( 36 ,-2, 0);
 Object(3) = sim_setObject( 36 ,4, 0);
 Object(4) = sim_setObject( 32 ,2, 0);
%=============================================
%                   Antenna Location
 load('Ap.mat');
simdata.AntX = Ap(:,1);
simdata.AntY = Ap(:,2);
simdata.AntZ = Ap(:,3);
% Get Phase History from simulation
simdata = sim_ConstructPhdata(simdata,Object);
%-------------------------------------------------------------------------------------
phdata = simdata.phdata';
save('IQ_data.mat', 'phdata')
% Test
% simdata.maxWr = c/(2*simdata.deltaF);
% simdata.r_vec = linspace(0,simdata.Nfft-1,simdata.Nfft)*simdata.maxWr/simdata.Nfft;
% %simdata.r_vec = linspace(-simdata.Nfft/2,simdata.Nfft/2-1,simdata.Nfft)*simdata.maxWr/simdata.Nfft;
```

Figure 20: Back-Projection Main function MATLAB code 1

```
% figure
% %stem( simdata.r_vec,abs(ifft(simdata.phdata(:,1),simdata.Nfft)));
% plot( simdata.r_vec,abs(ifft(simdata.phdata(:,1),simdata.Nfft)));
% xlabel('Distance (m)')
% ylabel('siginal intensity')
% hold on
%plot( simdata.r_vec, fftshift(abs(ifft(simdata.phdata(:,2),simdata.Nfft))));
% plot( simdata.r_vec, fftshift(abs(ifft(simdata.phdata(:,41),simdata.Nfft))));
% plot( simdata.r_vec, fftshift(abs(ifft(simdata.phdata(:,81),simdata.Nfft))));
% plot( simdata.r_vec, fftshift(abs(ifft(simdata.phdata(:,121),simdata.Nfft))));
% plot( simdata.r_vec, fftshift(abs(ifft(simdata.phdata(:,161),simdata.Nfft))));
% hold on
% plot(abs(fftshift(ifft(simdata.phdata(:,20),simdata.Nfft))));
% plot(abs(fftshift(ifft(simdata.phdata(:,40),simdata.Nfft))));
% plot(abs(fftshift(ifft(simdata.phdata(:,60),simdata.Nfft))));
%%
% Define the imaging grid using MATLAB function meshgrid
simdata.x_vec = linspace(simdata.CenterX-simdata.Wx/2,simdata.CenterX+simdata.Wx/2,simdata.Nx);
simdata.y_vec = linspace(simdata.CenterY-simdata.Wy/2,simdata.CenterY+simdata.Wy/2,simdata.Ny);
[simdata.x_mat,simdata.y_mat] = meshgrid(simdata.x_vec,simdata.y_vec);
simdata.z_mat = zeros(size(simdata.x_mat));
simdata.phdata=phdata';
%simdata.phdata2=phdata2';
simdata = bp(simdata);
subfinal1=simdata.im_final;
figure
imagesc(simdata.x_vec,simdata.y_vec,20*log10(abs(simdata.im_final)./...
    max(max(abs(simdata.im_final)))),[-dyn_range 0])
 axis xy image;
 h = xlabel('x (m)');
set(h,'FontSize',14,'FontWeight','Bold');
h = ylabel('y (m)');
set(h,'FontSize',14,'FontWeight','Bold');
 colorbar
toc


```

Figure 21: Back-Projection Main function MATLAB code 2

Section 6.2: Fast Back-Projection Main function MATLAB code

```
close all, clear, clc, format compact
c = 299792458; % Light speed
tic
simdata.BW = 400 * 10^6; % BandWidth
simdata.minF = 9.6 * 10^9; % Lowest Frequency of each pulse
simdata.K = 256; % Number of samples per sweep
% Frequency of the chirp
simdata.freq = linspace(simdata.minF, simdata.minF+simdata.BW, simdata.K);
simdata.deltaF = simdata.freq(2)-simdata.freq(1);
simdata.Nfft = 4096;
simdata.CenterX = 34.64;
simdata.CenterY = 0;
simdata.Wx = 10;
simdata.Wy = 10;
% simdata.Nx = 2002;
% simdata.Ny = 2002;
 simdata.Nx = 225;
 simdata.Ny = 225;
dyn_range = 5;              % dB of dynamic range to display
% Define the imaging grid for subimage using MATLAB function meshgrid
simdata.x_vec = linspace(simdata.CenterX-simdata.Wx/2,simdata.CenterX+simdata.Wx/2,simdata.Nx);
simdata.y_vec = linspace(simdata.CenterY-simdata.Wy/2,simdata.CenterY+simdata.Wy/2,simdata.Ny);
[simdata.x_mat,simdata.y_mat] = meshgrid(simdata.x_vec,simdata.y_vec);
simdata.z_mat = zeros(size(simdata.x_mat));
%=============================================
%                  Object Location
Object(1) = sim_setObject(32, 4, 0);
 Object(2) = sim_setObject( 36 ,-2, 0);
 Object(3) = sim_setObject( 36 ,4, 0);
 Object(4) = sim_setObject( 32 ,2, 0);
%=============================================
%                  Antenna Location
 load('Ap.mat');
 simdata2 = simdata;
    for n=1:1:80
        simdata.AntX(n)=Ap(2*n,1);
        simdata.AntY(n)=Ap(2*n,2);
        simdata.AntZ(n)=Ap(2*n,3);
```

Figure 22: Fast Back-Projection Main function MATLAB code 1

```
        simdata2.AntX(n)=Ap(2*n+1,1);
        simdata2.AntY(n)=Ap(2*n+1,2);
        simdata2.AntZ(n)=Ap(2*n+1,3);
    end
%=================================================
% Get Phase History from simulation
simdata = sim_ConstructPhdata(simdata,Object);
simdata2 =  sim_ConstructPhdata(simdata2,Object);
%%
simdata = bp(simdata);
subfinal1=simdata.im_final;
simdata2 = bp(simdata2);
subfinal2=simdata2.im_final;
im_final=ones(450);
i=1;
while i<451
        n=1;
    while n<451
        if  xor(mod(i,2),mod(n,2))  ==1
            im_final(i,n)=0;
        elseif mod(i,2)==1
             a=(i+1)/2;
             b=(n+1)/2;
             im_final(i,n)=subfinal1(a,a);
        else
            a=i/2;
            b=n/2;
            im_final(i,n)=subfinal2(a,b);
        end
        n=n+1;
    end
i=i+1;
end
  Nx = 450;
  Ny = 450;
 CenterX = 34.64;
 CenterY = 0;
 Wx = 10;
```

Figure 23: Fast Back-Projection Main function MATLAB code 2

```
 Wy = 10;
 % Define the imaging grid using MATLAB function meshgrid
 x_vec = linspace( CenterX- Wx/2, CenterX+ Wx/2, Nx);
 y_vec = linspace( CenterY- Wy/2, CenterY+ Wy/2, Ny);
 [x_mat, y_mat] = meshgrid( x_vec, y_vec);
 z_mat = zeros(size( x_mat));
figure
imagesc( x_vec, y_vec,20*log10(abs( im_final)./...
    max(max(abs( im_final)))),[-dyn_range 0])
 axis xy image;
 h = xlabel('x (m)');
set(h,'FontSize',14,'FontWeight','Bold');
h = ylabel('y (m)');
set(h,'FontSize',14,'FontWeight','Bold');
 colorbar
toc
```

Figure 24: Fast Back-Projection Main function MATLAB code 3

Section 6.3: New Fast Back-Projection Main function MATLAB code

```
close all, clear, clc, format compact
c = 299792458; % Light speed
simdata.BW = 400 * 10^6; % BandWidth
simdata.minF = 9.6 * 10^9; % Lowest Frequency of each pulse
simdata.K = 256; % Number of samples per sweep
simdata.freq = linspace(simdata.minF, simdata.minF+simdata.BW, simdata.K); % Frequency of the chirp
simdata.deltaF = simdata.freq(2)-simdata.freq(1);
simdata.Nfft = 4096;
simdata.CenterX = 34.64;
simdata.CenterY = 0;
simdata.Wx = 10;
simdata.Wy = 10;
% simdata.Nx = 2002;
% simdata.Ny = 2002;
 simdata.Nx = 700;
 simdata.Ny = 700;
dyn_range = 5;              % dB of dynamic range to display
 CenterX = 34.64;
 CenterY = 0;
 Wx = 10;
 Wy = 10;
%  Nx = 2002;
%  Ny = 2002;
  Nx = 450;
  Ny = 450;
% Define the imaging grid for subimage using MATLAB function meshgrid
 x_vec = linspace( CenterX- Wx/2, CenterX+ Wx/2, Nx);
 y_vec = linspace( CenterY- Wy/2, CenterY+ Wy/2, Ny);
[ x_mat, y_mat] = meshgrid( x_vec, y_vec);
 z_mat = zeros(size( x_mat));
 load('Ap.mat');
simdata.AntX = Ap(:,1);
simdata.AntY = Ap(:,2);
simdata.AntZ = Ap(:,3);

simdata2 = simdata;
size_mat=size(x_mat,1);
a=(size mat)/2;
```

Figure 25: New Fast Back-Projection Main function MATLAB code 1

```
b=(size_mat)/2;
sub1=ones(a);
sub2=ones(b);
n=1;
while n<(size_mat+1)
    i=1;
    while i<(size_mat+1)
        if and(mod(i,2)==1,mod(n,2)==1)
            simdata.x_mat((n+1)/2,(i+1)/2)=x_mat(n,i);
            simdata.y_mat((n+1)/2,(i+1)/2)=y_mat(n,i);

        else if and(mod(i,2)==0,mod(n,2)==0)

            simdata2.x_mat(n/2,(i)/2)=x_mat(n,i);
            simdata2.y_mat(n/2,(i)/2)=y_mat(n,i);
            end
        end
        i=i+1;
    end
    n=n+1;
end
simdata.z_mat=zeros(size(simdata.x_mat,1));
simdata2.z_mat=zeros(size(simdata2.x_mat,1));
%=============================================
%                   Object Location
Object(1) = sim_setObject(32, 4, 0);
 Object(2) = sim_setObject( 36 ,-2, 0);
 Object(3) = sim_setObject( 36 ,4, 0);
 Object(4) = sim_setObject( 32 ,2, 0);
% Get Phase History from simulation
simdata = sim_ConstructPhdata(simdata,Object);
simdata2 =  sim_ConstructPhdata(simdata2,Object);
%%
simdata = bp(simdata);
subfinal1=simdata.im_final;
simdata2 = bp(simdata2);
subfinal2=simdata2.im_final;
im_final=ones(450);
i=1;
```

Figure 26: New Fast Back-Projection Main function MATLAB code 2

```
while i<451
        n=1;
    while n<451
        if  xor(mod(i,2),mod(n,2))  ==1
            im_final(i,n)=0;
        elseif mod(i,2)==1
             a=(i+1)/2;
             b=(n+1)/2;
             im_final(i,n)=subfinal1(a,b);
        else
            a=i/2;
            b=n/2;
            im_final(i,n)=subfinal2(a,b);
        end
        n=n+1;
    end
i=i+1;
end
   Nx = 450;
   Ny = 450;
  CenterX = 34.64;
  CenterY = 0;
  Wx = 10;
  Wy = 10;
  % Define the imaging grid using MATLAB function meshgrid
  x_vec = linspace( CenterX- Wx/2, CenterX+ Wx/2, Nx);
  y_vec = linspace( CenterY- Wy/2, CenterY+ Wy/2, Ny);
  [x_mat, y_mat] = meshgrid( x_vec, y_vec);
  z_mat = zeros(size( x_mat));
 figure
 imagesc( x_vec, y_vec,20*log10(abs( im_final)./...
     max(max(abs( im_final)))),[-dyn_range 0])
  axis xy image;
  h = xlabel('x (m)');
 set(h,'FontSize',14,'FontWeight','Bold');
 h = ylabel('y (m)');
 set(h,'FontSize',14,'FontWeight','Bold');
  colorbar
 toc
```

Figure 27: New Fast Back-Projection Main function MATLAB code 3

Section 6.4: Sub-functions:

Back-projection function [5]:

```
function data = bp(data)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs a basic Backprojection operation.  The        %
% following fields need to be populated:                               %
%                                                                      %
% data.Nfft:  Size of the FFT to form the range profile                %
% data.deltaF:  Step size of frequency data (Hz)                       %
% data.minF:  The start frequency of each pulse (Hz) %
% data.x_mat:  The x-position of each pixel (m)                        %
% data.y_mat:  The y-position of each pixel (m)                        %
% data.z_mat:  The z-position of each pixel (m)                        %
% data.AntX:  The x-position of the sensor at each pulse (m)           %
% data.AntY:  The y-position of the sensor at each pulse (m)           %
% data.AntZ:  The z-position of the sensor at each pulse (m)           %
% data.phdata:  Phase history data (frequency domain)                  %
%               Fast time in rows, slow time in columns                %
%                                                                      %
% The output is:                                                       %
% data.im_final:  The complex image value at each pixel                %
%                                                                      %
% Written by LeRoy Gorham, Air Force Research Laboratory, WPAFB, OH    %
% Email:  leroy.gorham@wpafb.af.mil                                    %
% Date Released:  8 Apr 2010                                           %
%                                                                      %
% Gorham, L.A. and Moore, L.J., "SAR image formation toolbox for       %
%   MATLAB,"  Algorithms for Synthetic Aperture Radar Imagery XVII     %
%   7669, SPIE (2010).                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define speed of light (m/s)
c = 299792458;
% % Determine the size of the phase history data
% data.K = size(data.phdata,1);  % The number of frequency bins per pulse
% data.Np = size(data.phdata,2); % The number of pulses
 data.K = (size(data.phdata,1));  % The number of frequency bins per pulse
 data.Np = (size(data.phdata,2)); % The number of pulses
% Determine the azimuth angles of the image pulses (radians)
data.AntAz = unwrap(atan2(data.AntY,data.AntX));
% Determine the average azimuth angle step size (radians)
data.deltaAz = abs(mean(diff(data.AntAz)));
```

Figure 28: Back-Projection sub function 1

```
% Determine the total azimuth angle of the aperture (radians)
data.totalAz = max(data.AntAz) - min(data.AntAz);
% Determine the maximum scene size of the image (m)
data.maxWr = c/(2*data.deltaF);
data.maxWx = c/(2*data.deltaAz*mean(data.minF));
% Determine the resolution of the image (m)
data.dr = c/(2*data.deltaF*data.K);
data.dx = c/(2*data.totalAz*mean(data.minF));
% Display maximum scene size and resolution
fprintf('Maximum Scene Size:  %.2f m range, %.2f m cross-range\n',data.maxWr,data.maxWx);
fprintf('Resolution:  %.2fm range, %.2f m cross-range\n',data.dr,data.dx);
% Calculate the range to every bin in the range profile (m)
data.r_vec = linspace(-data.Nfft/2,data.Nfft/2-1,data.Nfft)*data.maxWr/data.Nfft;
%data.r_vec = linspace(0,data.Nfft,data.Nfft)*data.maxWr/data.Nfft;
% Initialize the image with all zero values
data.im_final = zeros(size(data.x_mat));
% Set up a vector to keep execution times for each pulse (sec)
t = zeros(1,data.Np);
% Loop through every pulse
for ii = 1:data.Np
    % Form the range profile with zero padding added
        rc = fftshift(ifft(data.phdata(:,ii),data.Nfft));
     %rc = ifft(data.phdata(:,ii),data.Nfft);
    % Calculate differential range for each pixel in the image (m)
    dR = sqrt((data.AntX(ii)-data.x_mat).^2 + ...
        (data.AntY(ii)-data.y_mat).^2 + ...
        (data.AntZ(ii)-data.z_mat).^2);
    % Calculate phase correction for image
    phCorr = exp(1i*4*pi*data.minF/c*dR);
    % Determine which pixels fall within the range swath
    I = find(and(dR > min(data.r_vec), dR < max(data.r_vec)));
    % Update the image using linear interpolation
    data.im_final(I) = data.im_final(I) + interp1(data.r_vec,rc,dR(I),'linear') .* phCorr(I);
     %data.im_final(I) = data.im_final(I) + interp1(data.r_vec,rc,dR(I),'linear');
    % Determine the execution time for this pulse
 %   t(ii) = toc;
end
return
```

Figure 29: Back-Projection sub function 2

Construct phase data function:

```
function simdata= sim_ConstructPhdata(simdata,Object)
            c = 299792458; % Light speed
            Np = length(simdata.AntX);
            Num_Object = length(Object);
            simdata.phdata = zeros(simdata.K,Np);
            for kk=1:1:Np % Loop through all antenna location
                    for pp = 1:1:Num_Object
                            dR = sqrt((simdata.AntX(kk)-Object(pp).X)^2+...
                            (simdata.AntY(kk)-Object(pp).Y)^2+...
                            (simdata.AntZ(kk)-Object(pp).Z)^2);
                            simdata.phdata(:, kk) = simdata.phdata(:,kk)+exp(-1i*4*pi*dR/c.*simdata.freq');
                    end
            end
end
```

Figure 30: Construct Phase data function

Set object function:

```
function Object =sim_setObject (X_Object,Y_Object, Z_Object)
            Object.X = X_Object;
            Object.Y = Y_Object;
            Object.Z = Z_Object;
end
```

Figure 31: Set Object function